以下例子主要在說明如何在純種 C 語言中實作出一個堆疊 (Stack),
至於堆疊的定義以及用途請參考計概課本或是資料結構課本,
或者用 google.com 找一找應該也可看到許多實例與說明

```
tsaiwn@magpie % cat -n stk.c
     1  // stk.c
     2  // 注意何以變數要 static ?
     3  //#define NELEMENT 99
     4     enum{NELEMENT=99};     // better
     5     static int data[NELEMENT];    // why static ?
     6     static int stkptr = -1;
     7  // 以下三個函數宣告可有可無
     8        void push(int);
     9        int pop(void);
    10        int top(void);
    11  //////
    12  void push(int x) {
    13      stkptr++;
    14      data[stkptr] = x;
    15  }
    16  int top(void) {
    17      return data[stkptr];
    18  }
    19  int pop(void) {
    20      if(stkptr> -1) --stkptr;
    21  }
    22  int empty(void){ return (stkptr <= -1) ;}
    23  int isfull(void){ return (stkptr >= NELEMENT-1);}
tsaiwn@magpie %
tsaiwn@magpie % cat -n stkmain.c
     1  // stkmain.c  --- 使用 stk.c 中的堆疊 (Stack)
     2  // gcc stkmain.c stk.c   ;  ./a.out
     3  // 問題考慮: 如果需要兩個 stack 要怎麼做?
     4  // 答案: 較簡單方法是把stk.c複製到另一file, 修改function name例如push2/pop2
     5  // 因爲各file中變數有 static 保護, 不會被其它 file 中的 function 看到!
     6  #include<stdio.h>
     7  int main( ){
     8     push(880);
     9     push(770);
    10     push(53);
    11     while(!empty()){
    12        printf("%d ", top( ) );
    13        pop();
    14     }
    15     printf("\n");
    16  }
tsaiwn@magpie %
tsaiwn@magpie % gcc stkmain.c stk.c
tsaiwn@magpie % ./a.out
53 770 880

tsaiwn@magpie % exit
exit
```

用 **class** 把 **stack** 有關的 **data** 與 **function** 封藏起來!
以下連測試的主程式也寫一起是偷懶的做法!

```
tsaiwn@magpie% cat -n mystk.cpp
     1  //Object Oriented Programming features:
     2  //  Encapsulation
     3  //  Information hiding
     4  //  Inheritance
     5  //  Polymorphism
     6  //
     7  //
     8  class MyStack{
     9     public:
    10        enum{NELEMENT=99};
    11     private:
    12        int data[NELEMENT];
    13        int stkptr;
    14     public:
    15        MyStack(void){ stkptr= -1; }
    16        void push(int);
    17        int pop(void);
    18        int top(void);
    19        int empty(void){ return (stkptr <= -1) ;}
    20        int isfull(void){ return (stkptr >= NELEMENT-1);}
    21  };
    22
    23  void MyStack::push(int x)
    24  {
    25      stkptr++;
    26      data[stkptr] = x;
    27  }
    28  inline int MyStack ::top(void)
    29  {
    30      return data[stkptr];
    31  }
    32  inline int MyStack ::pop(void)
    33  {
    34      if(stkptr> -1) --stkptr;
    35  }
    36  //
    37  #include<iostream.h>
    38  //#include<stack.h>
    39  int main( ){
    40     //stack <int> x;
    41     MyStack x;
    42     x.push(880);
    43     x.push(770);
    44     x.push(53);
    45     while(!x.empty()){
    46        cout << x.top();  x.pop();
    47     }
    48     cout << endl;
    49  }
    50  // 此例是把 stack 與 主程式寫在一個檔案, 這是偷懶且不好的方法
    51  // 正確方式應該把stack獨立, 且應把宣告寫在 .h 檔案, 實作寫 .cpp檔案

tsaiwn@magpie% g++ mystk.cpp
tsaiwn@magpie% ./a.out
53770880
tsaiwn@magpie% exit
```

這是正確的寫法: 把宣告寫在 mystk2.h 這 header file,
  實作堆疊的程式碼則寫到 mystk2.cpp 中,
不過, 此例中仍留了兩個函數 empty( ) 與 isfull( ) 在 .h 檔案中,
因爲它們是直接寫在 class 內, 會自動變成 inline function. (會影響翻譯方式)

此外, 要注意不但在 mystk2.cpp 中要 #include "mystk2.h"
而且使用此堆疊的主程式之檔案也要 #include "mystk2.h"

```
tsaiwn@magpie % cat -n mystk2.h
     1  //mystk2.h
     2  class MyStack{
     3     public:
     4        enum{NELEMENT=99};
     5     private:
     6        int data[NELEMENT];
     7        int stkptr;
     8     public:
     9        MyStack(void){ stkptr= -1; }
    10        void push(int);
    11        int pop(void);
    12        int top(void);
    13        int empty(void){ return (stkptr <= -1) ;}
    14        int isfull(void){ return (stkptr >= NELEMENT-1);}
    15  };
tsaiwn@magpie % cat -n mystk2.cpp
     1  // mystk2.cpp -- implementation of the stack
     2  #include "mystk2.h"
     3  void MyStack::push(int x)
     4  {
     5      stkptr++;
     6      data[stkptr] = x;
     7  }
     8  int MyStack :: top(void)
     9  {
    10      return data[stkptr];
    11  }
    12  //Actually,  pop( ) function is a void function in C++ STL
    13  int MyStack ::pop(void)
    14  {
    15      if(stkptr> -1) --stkptr;
    16  }
tsaiwn@magpie % cat -n mymain2.cpp
     1  // mymain2.cpp;  g++ mymain2.cpp mystk2.cpp ; ./a.out
     2  #include "mystk2.h"
     3  //注意以下 iostream.h 爲舊的寫法; 1999之後的 C++ 使用 <iostream>
     4  #include<iostream.h>
     5  int main( ){
     6     MyStack x;
     7     x.push(880);
     8     x.push(770);
     9     x.push(53);
    10     while(!x.empty()){
    11        cout << x.top();  x.pop();
    12     }
    13     cout << endl;
    14  }
tsaiwn@magpie % g++ mymain2.cpp mystk2.cpp -Wno-deprecated
tsaiwn@magpie % ./a.out
53770880

tsaiwn@magpie % exit
exit
```

這範例是把之前的 STACK 範例重新寫成 template class(樣版類別);
寫成 template 時只寫成一個 .h 檔; 寫成非template 則會分 .h(宣告) 與 .cpp(實作);
注意以下 template class 寫法! 除了在 class 之前要寫 template < class T> 之外,
所有寫在 class { }; 結構之外的 class 的左邊(前面)也都要寫 template<class T>
而且在 :: 之左也要寫 <class T> ==> 例如 : (還有該換的 type 別忘了換爲 T)
template <class T> T MyStack<T> :: top(void) { ... } // 注意以下 T 寫成 GG

```
tsaiwn@magpie % cat -n mystk3.h
     1  //mystk3.h
     2  //a template class for a STACK, CopyLeft by tsaiwn@csie.nctu.edu.tw
     3  template <class GG>
     4  class MyStack{
     5     public:
     6        enum{NELEMENT=99};
     7     private:
     8        GG data[NELEMENT];
     9        int stkptr;
    10     public:
    11        MyStack(void){ stkptr= -1; }
    12        void push(GG  x);
    13        void pop(void);
    14        GG top(void);
    15        int empty(void){ return (stkptr <= -1) ;}
    16        int isfull(void){ return (stkptr >= NELEMENT-1);}
    17  };
    18
    19  template <class GG>
    20  GG MyStack<GG> :: top(void)
    21  {   // 注意上面兩列的寫法 !
    22      return data[stkptr];
    23  }
    24
    25  template <class GG>
    26  void MyStack<GG> :: push(GG  x) { data[++stkptr] = x; }
    27
    28  //Actually,  pop( ) function is a void function in C++ STL
    29  template <class GG>
    30  void MyStack<GG> ::pop(void) { if(stkptr > -1)  --stkptr; }
tsaiwn@magpie % cat -n mymain3.cpp
     1  // mymain3.cpp;  g++ mymain3.cpp   ; ./a.out
     2  #include "mystk3.h"
     3  //注意以下 iostream.h 爲舊的寫法; 1999之後的 C++ 使用 <iostream>
     4  #include<iostream.h>
     5  int main( ){
     6     MyStack <int> x;
     7     MyStack <double> y;
     8     x.push(880);
     9     x.push(770);
    10     x.push(53);
    11     while(!x.empty()){
    12        cout << x.top();  y.push( x.top() + 0.5 );
    13        x.pop();
    14     }
    15     cout << endl << "=== === now dump y:\n";
    16     while(!y.empty()){
    17        cout << y.top() << " "; y.pop();
    18     }
    19     cout << "\n=== bye bye ===" << endl;
    20  }
tsaiwn@magpie % g++ mymain3.cpp -Wno-deprecated
tsaiwn@magpie % ./a.out
53770880
=== === now dump y:
880.5 770.5 53.5
=== bye bye ===
tsaiwn@magpie % exit
exit
```

其實, C++ Class library 已經有 stack,
是以 template (樣板) 方式定義; 使用上更爲方便!
但其函數名稱與用法與一般資料結構課本上講的稍有不同;
例如, 一般pop不是void function. 又如是否空堆疊用 empty( ) 不是 isempty( )
因爲C++ 的一些資料結構class來自HP公司的 STL (Standard Template Library),
在更早時要 #include <stl.h> 才能使用 STL 裡面的 class.
後來, STL在1999年正式被納入 C++ 類別程式庫中, 可 include 各自的 header即可

```
tsaiwn@magpie % cat -n stk2.cpp
     1  //stk2.cpp   // g++ stk2.cpp ; ./a.out
     2  // 注意, 以下是用舊的寫法
     3  #include<iostream.h>
     4  #include<stack.h>
     5  int main( ) {
     6     stack <int> x;    // STL 中的 stack 是 template
     7     x.push(880);
     8     x.push(770);
     9     x.push(53);
    10     while(!x.empty()){
    11        cout << x.top();  x.pop();
    12     }
    13     cout << endl;
    14  }
tsaiwn@magpie % g++ stk2.cpp
 warning: #warning This file includes at least one deprecated or
     antiquated header. ...  To disable this warning use -Wno-deprecated.
tsaiwn@magpie % ./a.out
53770880

tsaiwn@magpie % cat -n stk3.cpp
     1  //stk3.cpp
     2  // g++ stk3.cpp ; ./a.out
     3  #include<iostream>
     4  #include<stack>
     5  // 新寫法沒有 .h 但要記得 using namespace std;
     6  using namespace std;  // 因爲C++ class library被集合到命名空間 std::
     7  int main( ){
     8     stack <int> x;    // STL 中的 stack 是 template
     9     x.push(880);
    10     x.push(770);
    11     x.push(53);
    12     while(!x.empty()){
    13        cout << x.top() << " ";  x.pop();
    14     }
    15     cout << endl;
    16  }
tsaiwn@magpie % g++ stk3.cpp
tsaiwn@magpie % g++ ./a.out
53 770 880
tsaiwn@magpie % exit
exit

tsaiwn@magpie %
tsaiwn@magpie % cat /usr/include/g++/stl.h
// -*- C++ -*- compatibility header.
// This file is part of the GNU ANSI C++ Library.

#include <algorithm>
#include <deque>
#include <functional>
#include <iterator>
#include <list>
#include <map>
#include <memory>
#include <numeric>
#include <set>
#include <stack>
#include <utility>
#include <vector>
tsaiwn@magpie %
```

```
ccbsd5% cat -n inline.cpp
     1  // inline.cpp -- @CopyLeft by tsaiwn@csie.nctu.edu.tw
     2  // g++ -c inline.cpp; nm inline.o | grep my
     3  #ifndef __GNUG__
     4   #include<iostream.h>
     5   // possibly in Torbo C 3.x
     6  #else
     7   #include <iostream>
     8   using namespace std;
     9  #endif
    10  // myfff, myggg, myhhh, mymmm 都是做一樣的事,
    11  // 但 myggg 與 myhhh 爲 inline, myhhh因沒用到就沒在 .o 檔案中!
    12  // 另外, myhhh 和 mymmm 都沒用到, 但 myhhh 爲 inline function,
    13  // 翻譯出的機器碼中找不到 myhhh, 但仍有不是 inline 的 mymmm
    14  // 主程式中叫用 myfff 與 myggg 次數都是4次.
    15  int myfff(int x) {
    16     static int kkk=0;
    17     kkk += x*x;
    18     return kkk;
    19  }
    20  inline
    21   int myggg(int x) {
    22     static int kkk=0;
    23     kkk += x*x;
    24     return kkk;
    25  }
    26  inline
    27   int myhhh(int x) {
    28     static int kkk=0;
    29     kkk += x*x;
    30     return kkk;
    31  }
    32  int mymmm(int x) {  // 這不是 inline, 沒用到仍會在 .o 檔案中!
    33     static int kkk=0;
    34     kkk += x*x;
    35     return kkk;
    36  }
    37  int main(){
    38     cout << myfff(1) << endl;
    39     cout << myfff(2) << endl;
    40     cout << myfff(3) << endl;
    41     cout << myfff(4) << endl;
    42     cout << myfff(5) << endl;
    43     cout << "===" << endl;
    44     cout << myggg(1) << endl;
    45     cout << myggg(2) << endl;
    46     cout << myggg(3) << endl;
    47     cout << myggg(4) << endl;
    48     cout << myggg(5) << endl;
    49     cout << "=== ===" << endl;
    50  }

ccbsd5% g++ -c inline.cpp
ccbsd5%
ccbsd5% nm inline.o | grep my
000003c2 t _GLOBAL__D__Z5myfffi
000003a8 t _GLOBAL__I__Z5myfffi
000000fc T _Z5myfffi
00000000 W _Z5mygggi
00000114 T _Z5mymmmi            這是 mymmm(int);
00000004 b _ZZ5myfffiE3kkk       <--- 這是 myfff 內的 static 變數 kkk
00000000 V _ZZ5mygggiE3kkk       <--- 這是 myggg 內的 static 變數 kkk
00000008 b _ZZ5mymmmiE3kkk       <--- 這是 mymmm 內的 static 變數 kkk
ccbsd5%
```